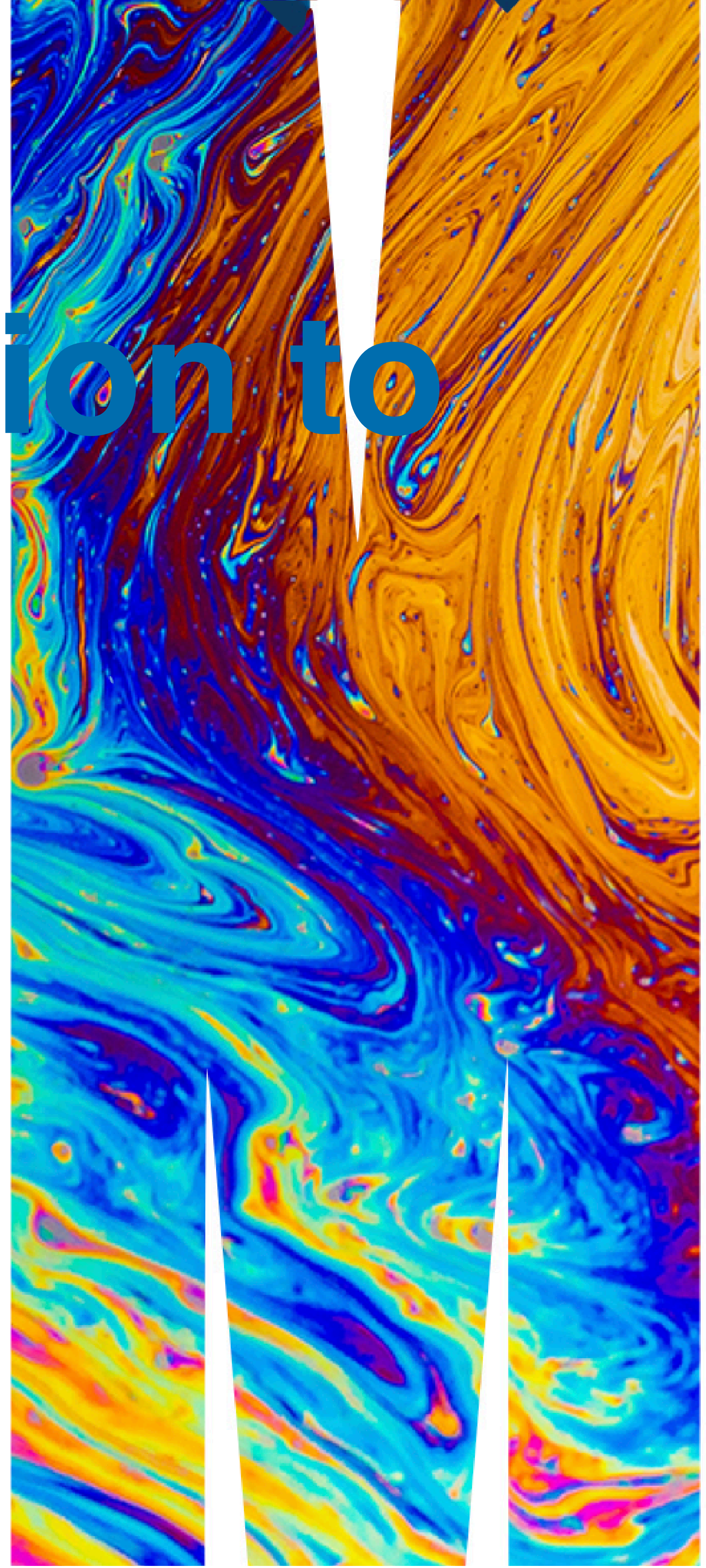
ETC3250/5250 Introduction to Machine Learning

Week 6: Neural networks and deep learning

Professor Di Cook

etc3250.clayton-x@monash.edu

Department of Econometrics and Business Statistics



Overview

We will cover:

- Structure of a neural network
- Fitting neural networks
- Diagnosing the fit

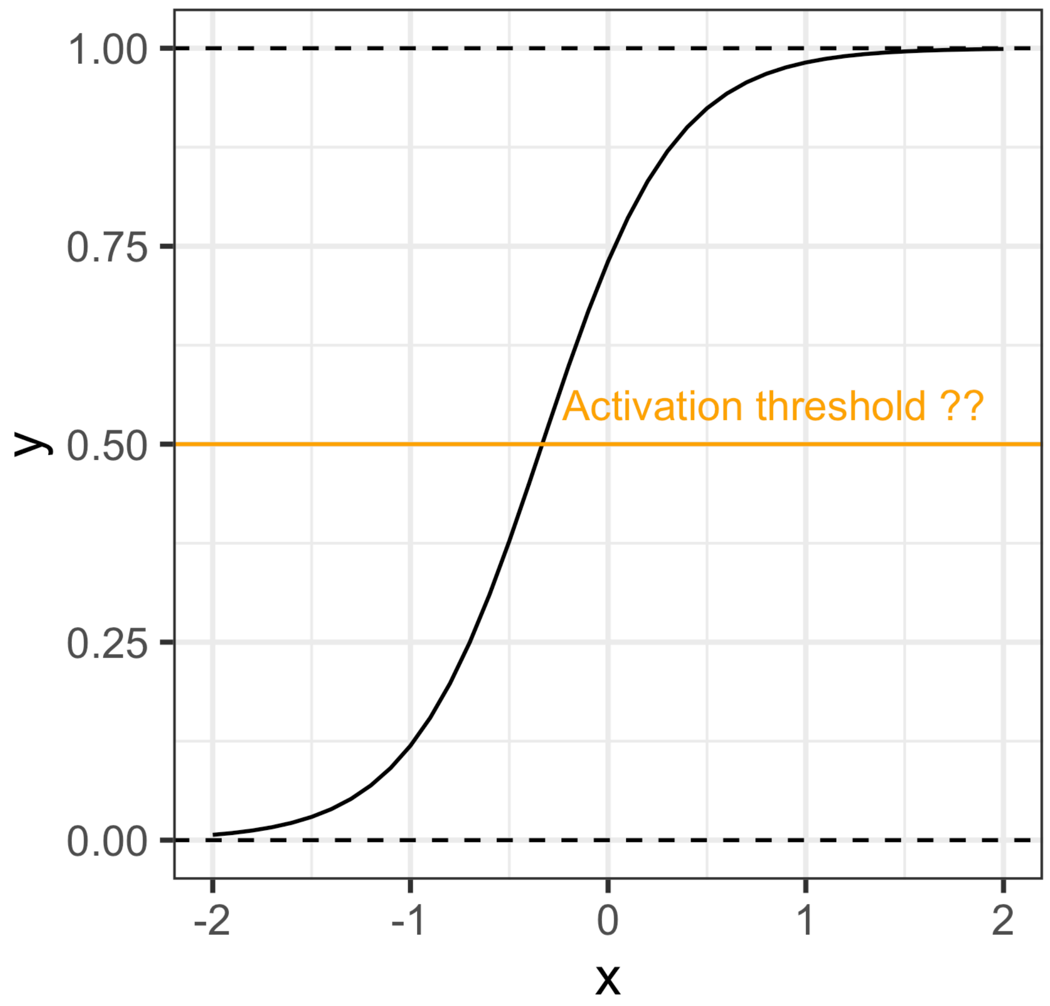
Structure of a neural network

Nested logistic regressions

Remember the logistic function:

Also,

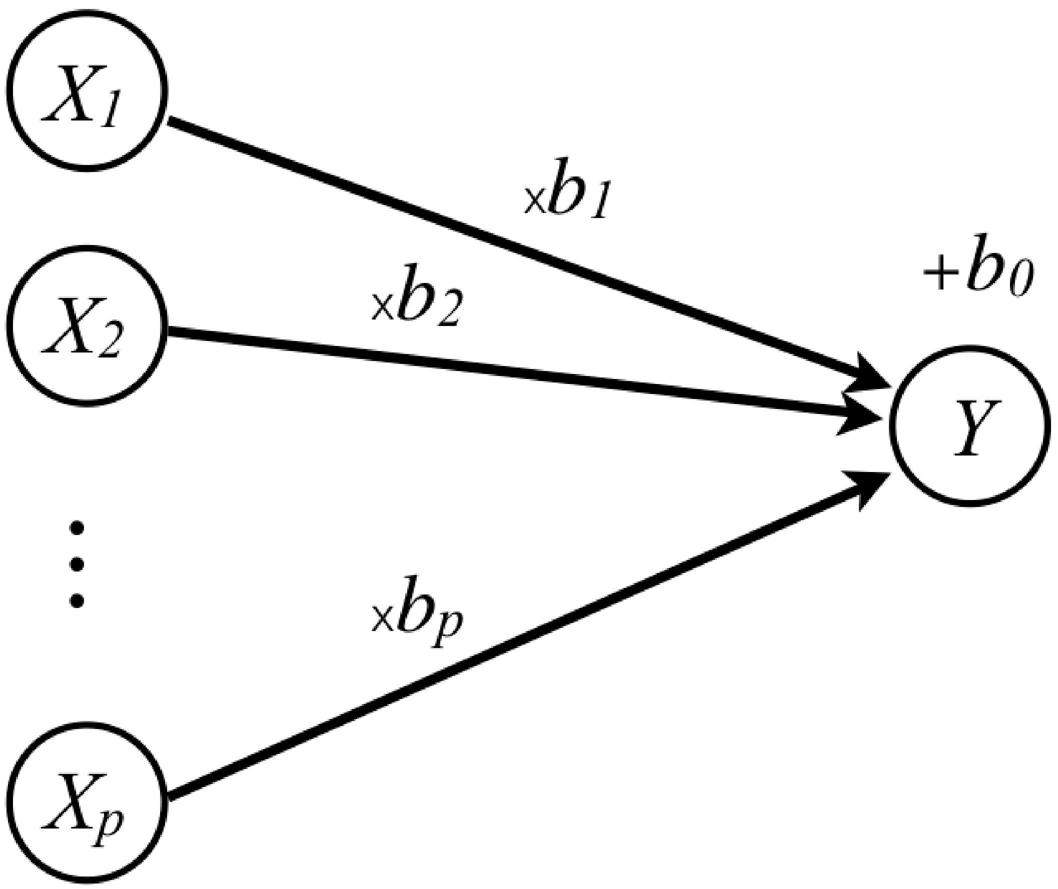
Above the threshold predict to be 1.



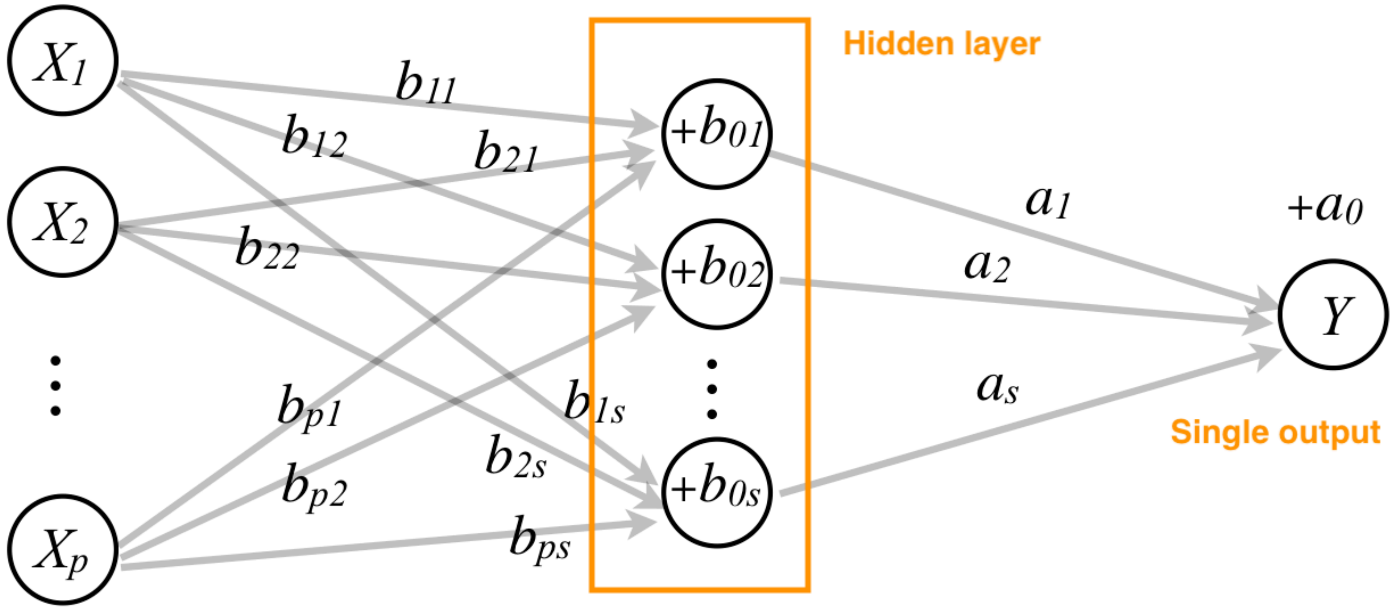
Linear regression as a network

Drawing as a network model:

inputs (predictors), multiplied by weights (coefficients), summed, add a constant, predicts output (response).

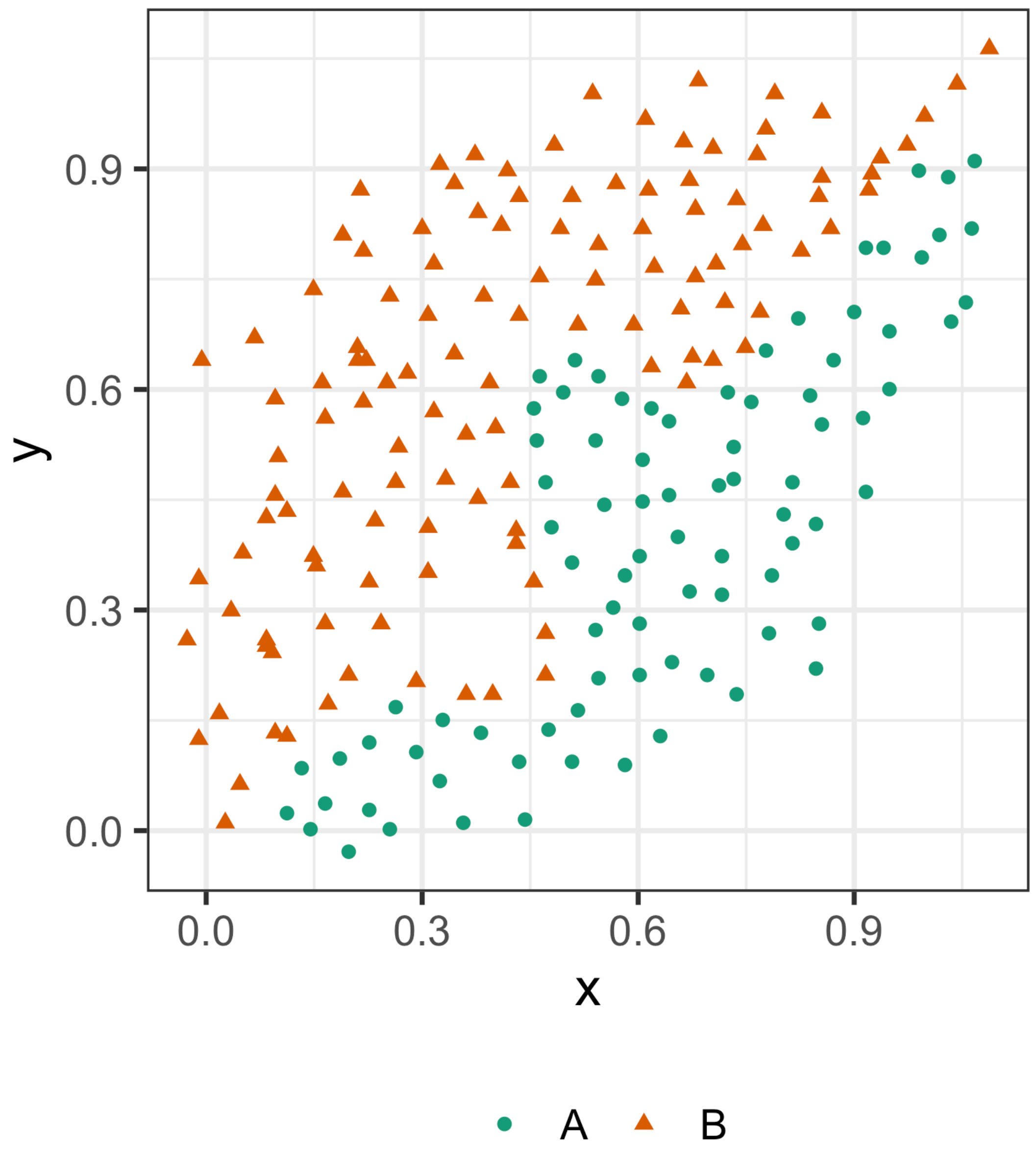


Single hidden layer NN

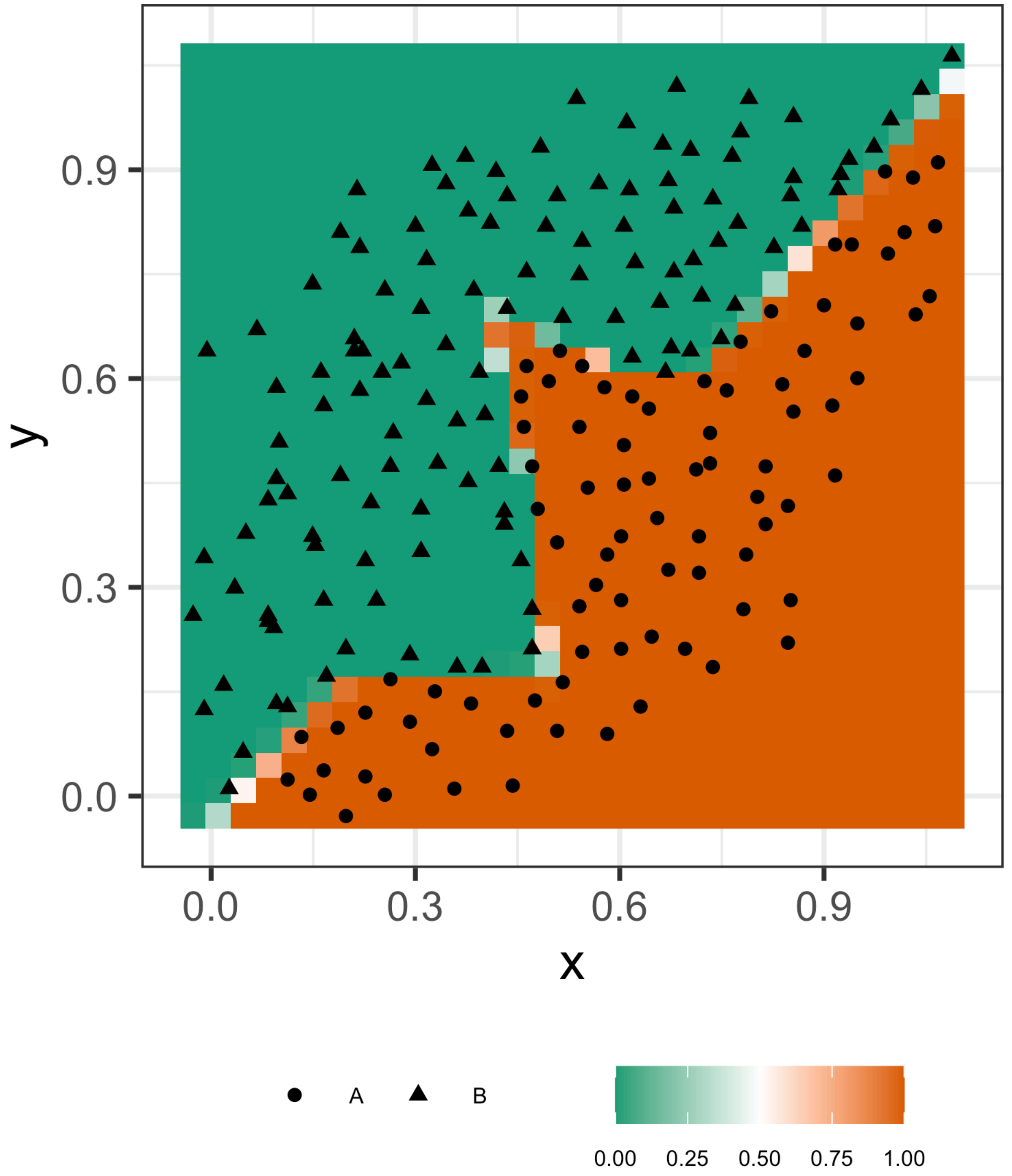


What does this look like? (1/2)

The architecture allows for combining multiple linear models to generate non-linear classifications.



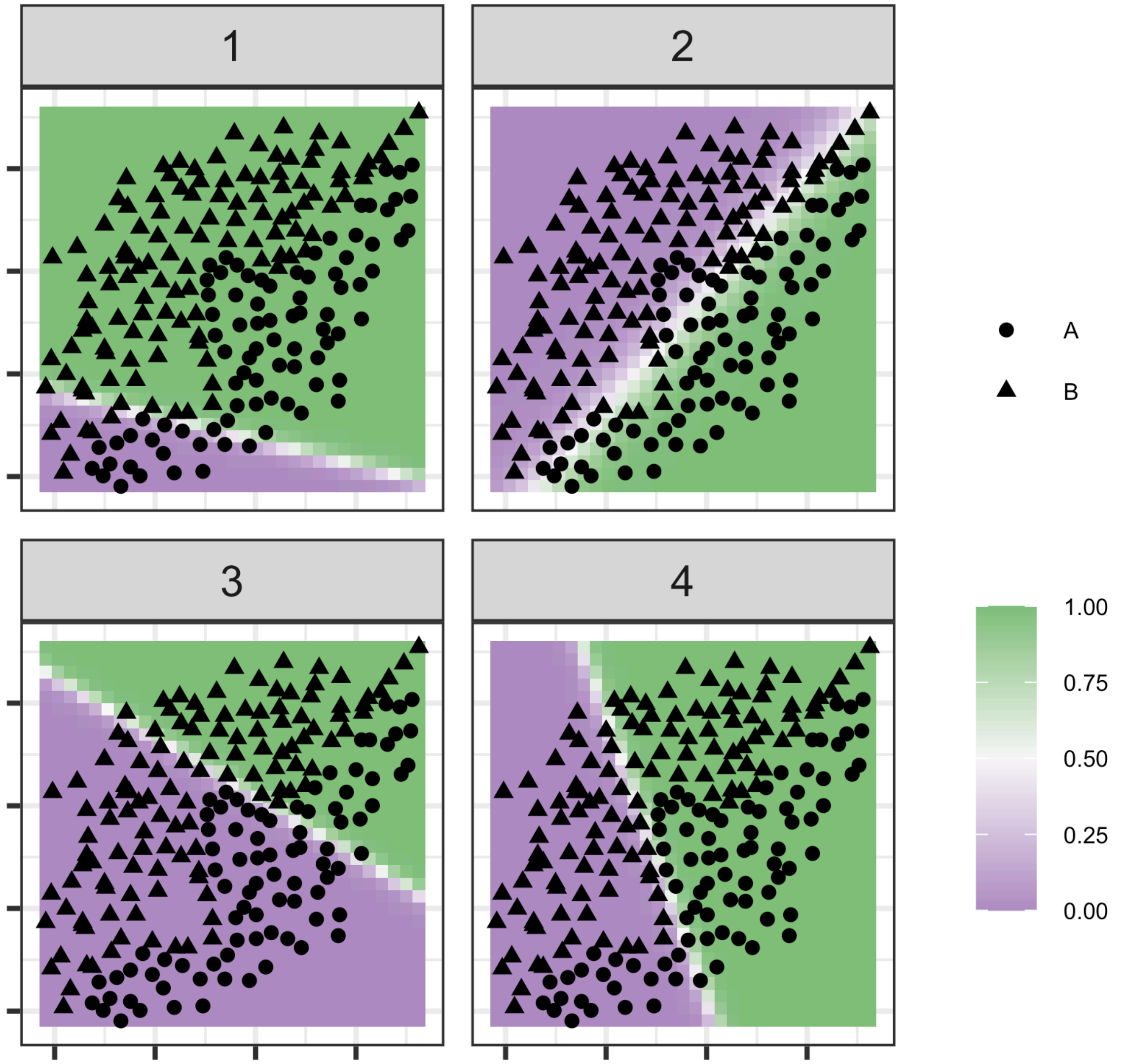
The best fit uses , four nodes in the hidden layer. Can you sketch four lines that would split this data well?

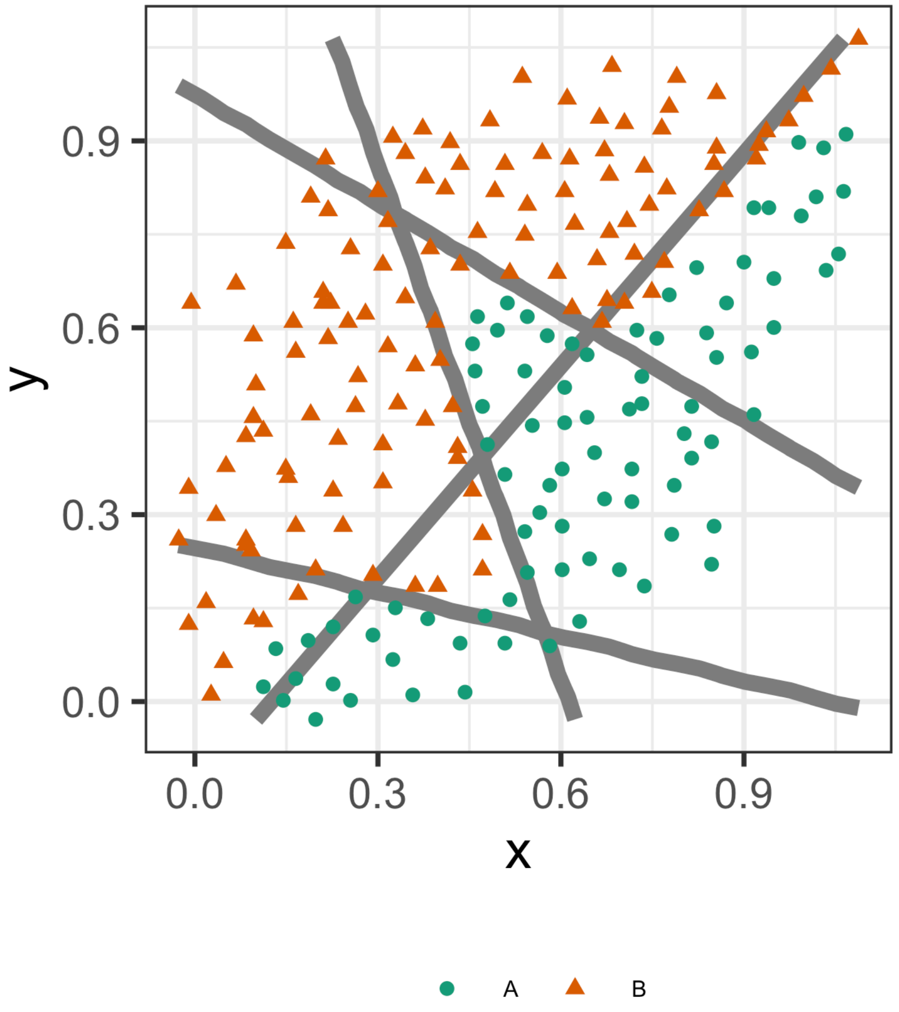


Wickham et al (2015) Removing the Blindfold

What does this look like? (2/2)

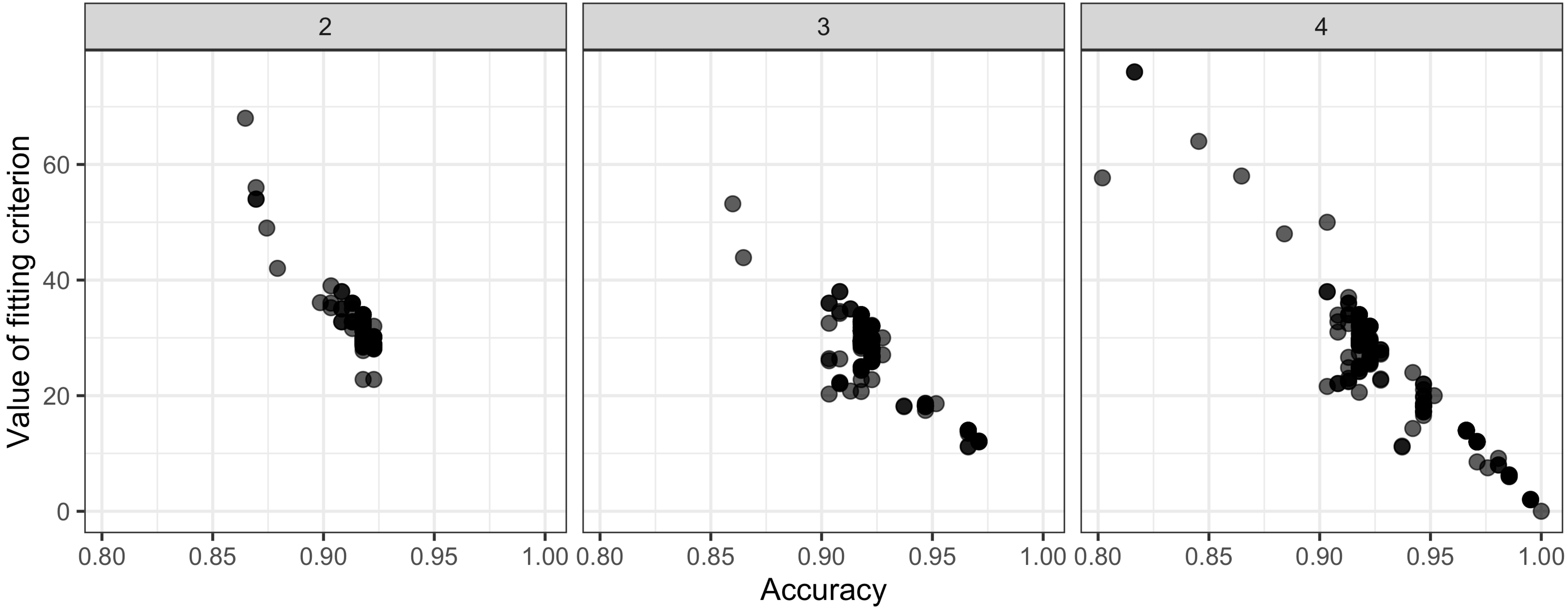
The models at each of the nodes of the hidden layer.





But can be painful to find the best!

These are all the models fitted, using with the fit statistics.



Fitted using the R package `nnet`. It's very unstable, and this is still a problem with current procedures.

Fitting with keras

Steps (1/2)

1. Define architecture
    - flatten: if you are classifying images, you need to flatten the image into a single row of data, eg 24x24 pixel image would be converted to a row of 576 values. Each pixel is a variable.
    - How many hidden layers do you need?
    - How many nodes in the hidden layer?
    - Dropout rate: proportion of nodes removed randomly at each update, for regularisation, to reduce number of parameters to be estimated

2. Specify activation: linear, relu (rectified linear unit), sigmoid, softmax

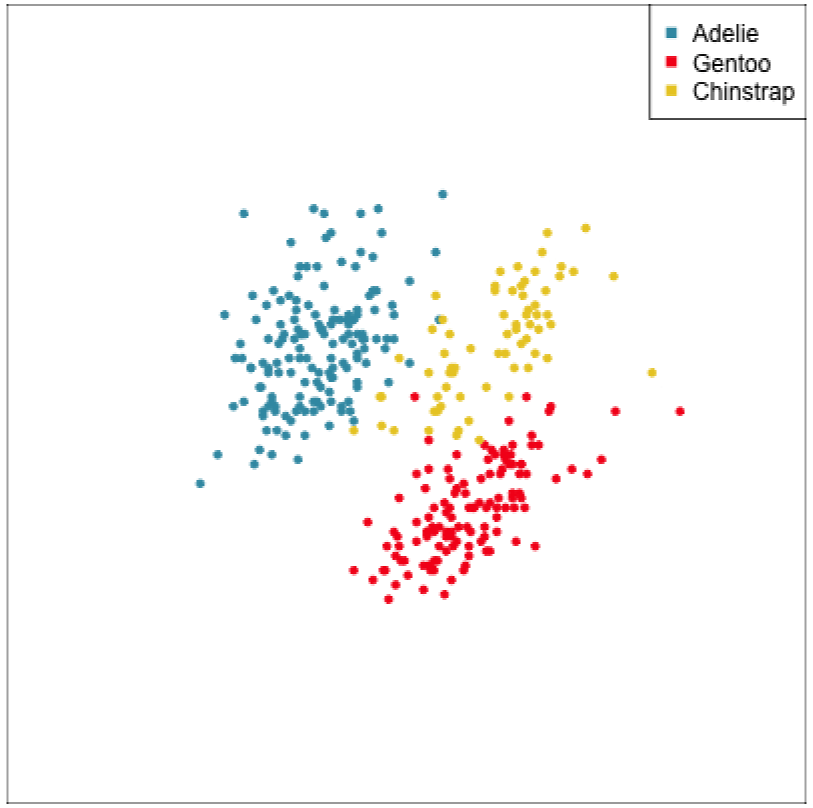
3. Choose loss function:
    - MSE: differencing predictive probabilities from binary matrix specified response, eg predict=(0.91, 0.07, 0.02) and true=(1,0,0) then loss is (1-0.91)^2=0.0081.
    - cross-entropy: where is true, and is predicted, eg -1 log_e(0.91)=0.094

Steps (2/2)

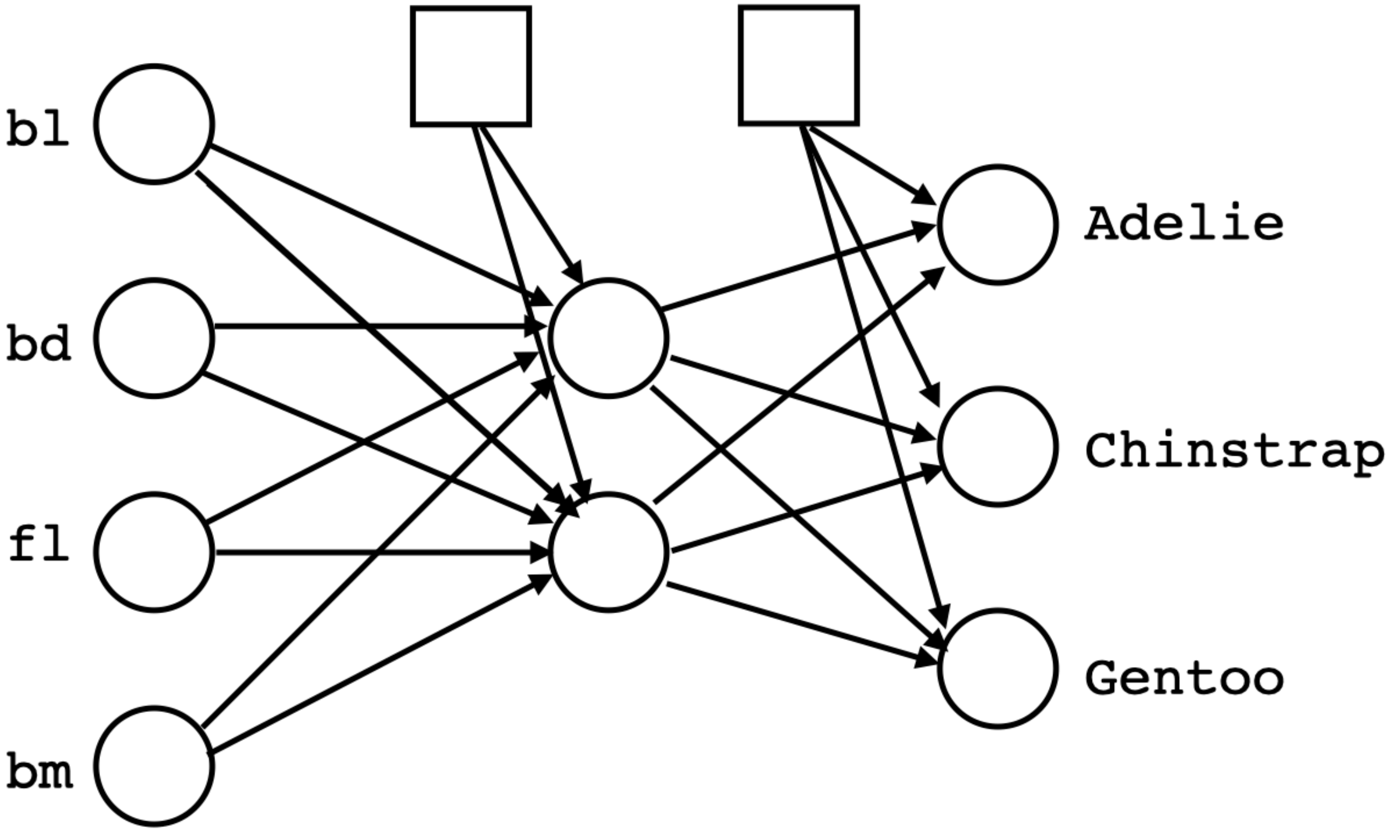
4. Training process:
   - epochs: number of times the algorithm sees the entire data set
   - batch_size: subset used at each fit
   - validation_split: proportion for hold-out set for computing error rate
   - batch_normalization: each batch is standardised during the fitting, can be helpful even if full data is standardised

5. Evaluation:
   - usual metrics: accuracy, ROC, AUC (area under ROC curve), confusion table
   - visualise the predictive probabilities
   - examine misclassifications
   - examine specific nodes, to understand what part it plays
   - examine model boundary relative to the observed data

Example: penguins (1/5)



Choose 2 nodes, because reducing to 2D, like LDA discriminant space, makes for easy classification.



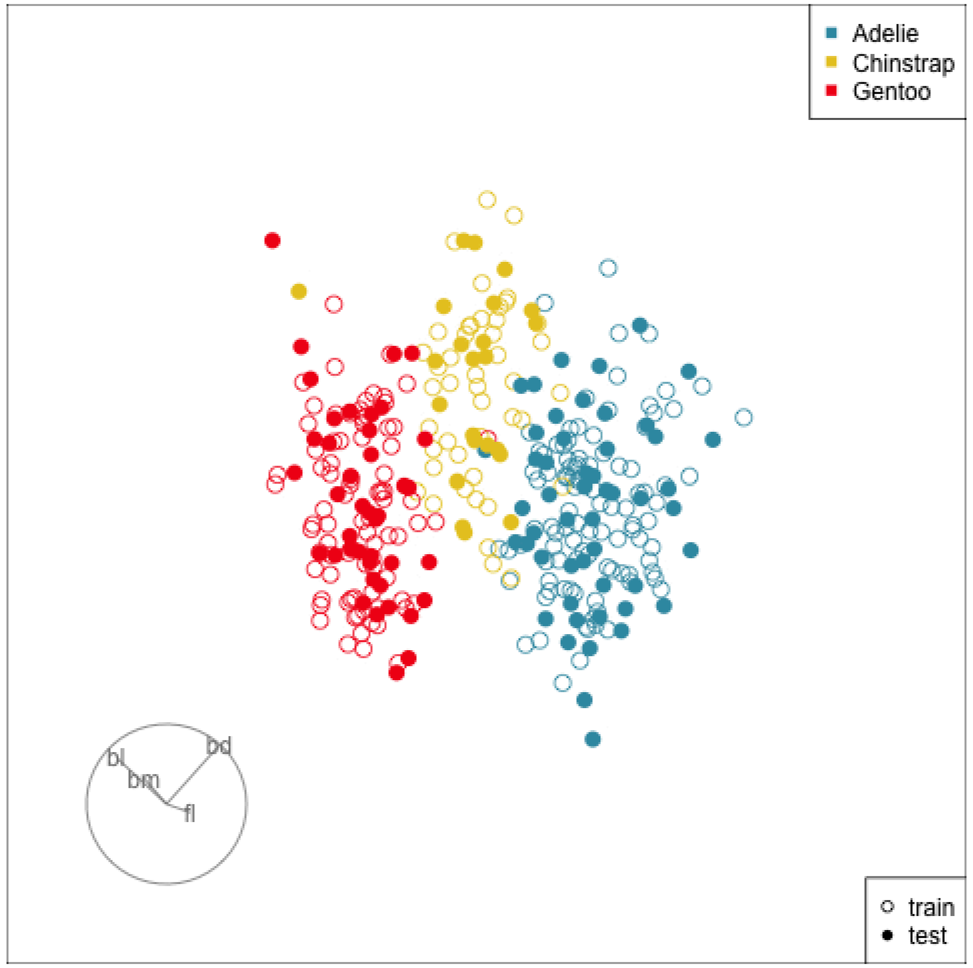
- 4D data
- Simple cluster structure to classes
- How many nodes in the hidden layer?

Example: penguins (2/5)

```
library(keras)
tensorflow::set_random_seed(211)

# Define model
p_nn_model <- keras_model_sequential()
p_nn_model %>%
  layer_dense(units = 2, activation = 'relu',
              input_shape = 4) %>%
  layer_dense(units = 3, activation = 'softmax
p_nn_model %>% summary

loss_fn <- loss_sparse_categorical_crossentrop
  from_logits = TRUE)

p_nn_model %>% compile(
  optimizer = "adam",
  loss      = loss_fn,
  metrics   = c('accuracy')
)
```

Note that the `tidymodels` code style does not allow easy extraction of model coefficients.

Split the data into training and test, and check it.



Example: penguins (3/5)

Fit the model

```
# Data needs to be matrix, and response needs t
p_train_x <- p_train %>%
  select(bl:bm) %>%
  as.matrix()
p_train_y <- p_train %>% pull(species) %>% as.n
p_train_y <- p_train_y-1 # Needs to be 0, 1, 2
p_test_x <- p_test %>%
  select(bl:bm) %>%
  as.matrix()
p_test_y <- p_test %>% pull(species) %>% as.nu
p_test_y <- p_test_y-1 # Needs to be 0, 1, 2
```

```
# Fit model
p_nn_fit <- p_nn_model %>%
  keras::fit(
    x = p_train_x,
    y = p_train_y,
    epochs = 200,
    verbose = 0
  )
```

How many parameters need to be estimated?

Four input variables, two nodes in the hidden layer and a three column binary matrix for output. This corresponds to 5+5+3+3+3=19 parameters.

```
Model: "sequential"
________________________________________________________________
 Layer (type)                Output Shape              Param #
================================================================
 dense_1 (Dense)             (None, 2)                 10
 dense (Dense)               (None, 3)                 9
================================================================
Total params: 19 (76.00 Byte)
Trainable params: 19 (76.00 Byte)
Non-trainable params: 0 (0.00 Byte)
________________________________________________________________
```

Example: penguins (4/5)

Evaluate the fit

```
p_nn_model %>%
  evaluate(p_test_x, p_test_y, verbose = 0)
```

```
    loss accuracy
    0.28     0.94
```

Confusion matrices for training and test

```
           p_train_pred_cat
            Adelie Chinstrap Gentoo
  Adelie        95         5      0
  Chinstrap      0        45      0
  Gentoo         1         0     81

           p_test_pred_cat
            Adelie Chinstrap Gentoo
  Adelie        46         3      2
  Chinstrap      0        23      0
  Gentoo         2         0     39
```

Note: Specifically have chosen settings so fit is not perfect

Estimated parameters

```
# Extract hidden layer model weights
p_nn_wgts <- keras::get_weights(p_nn_model, tr
p_nn_wgts
```

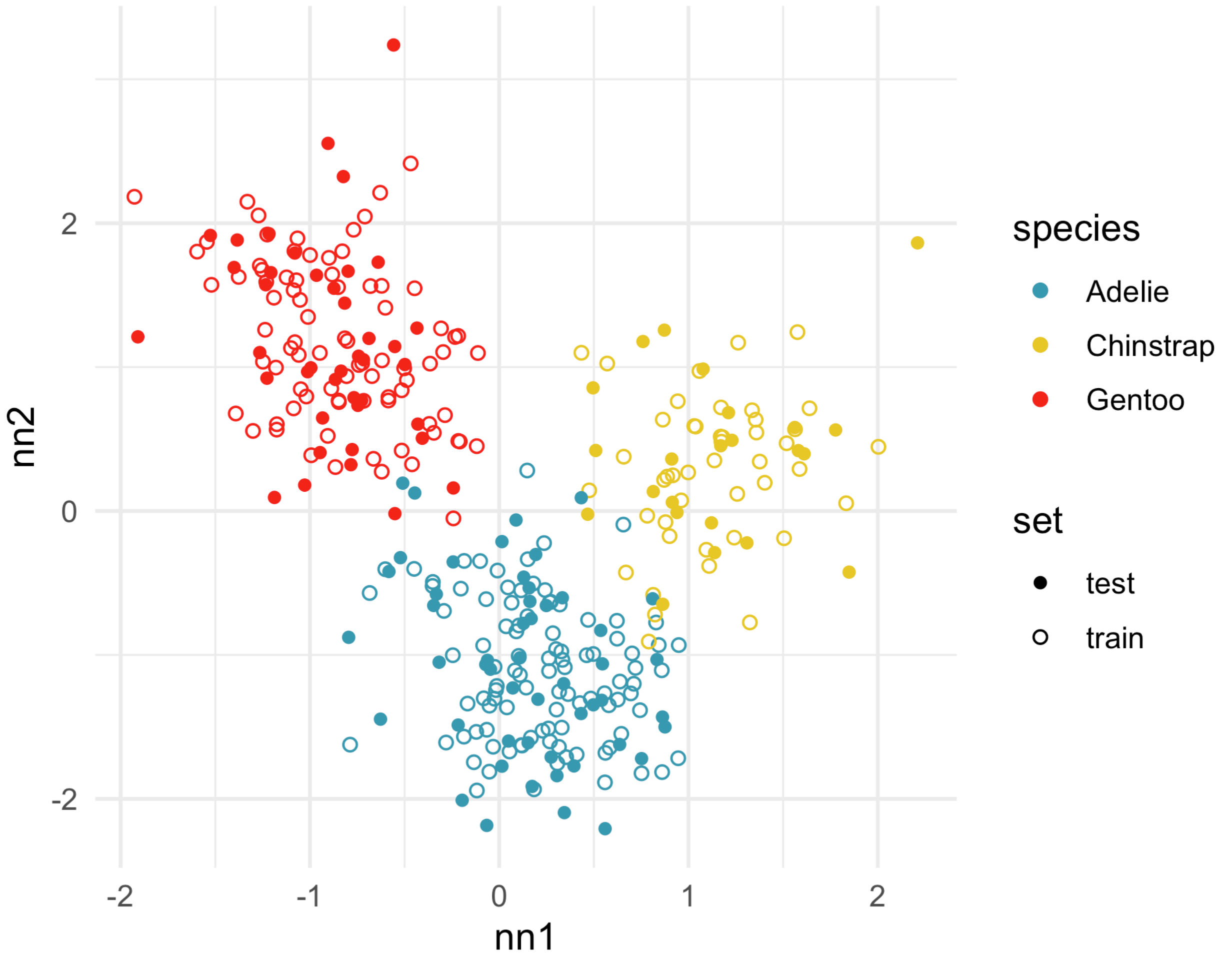
```
[[1]]
      [,1]   [,2]
[1,]  0.62  1.333
[2,]  0.19 -0.016
[3,] -0.17 -0.304
[4,] -0.89 -0.366

[[2]]
[1]  0.127 -0.095

[[3]]
      [,1] [,2]  [,3]
[1,] -0.16  1.5 -1.92
[2,] -0.75  1.6  0.32
```

Which variables are contributing most to each hidden layer node?

Can you write out the model?

Example: penguins (5/5)

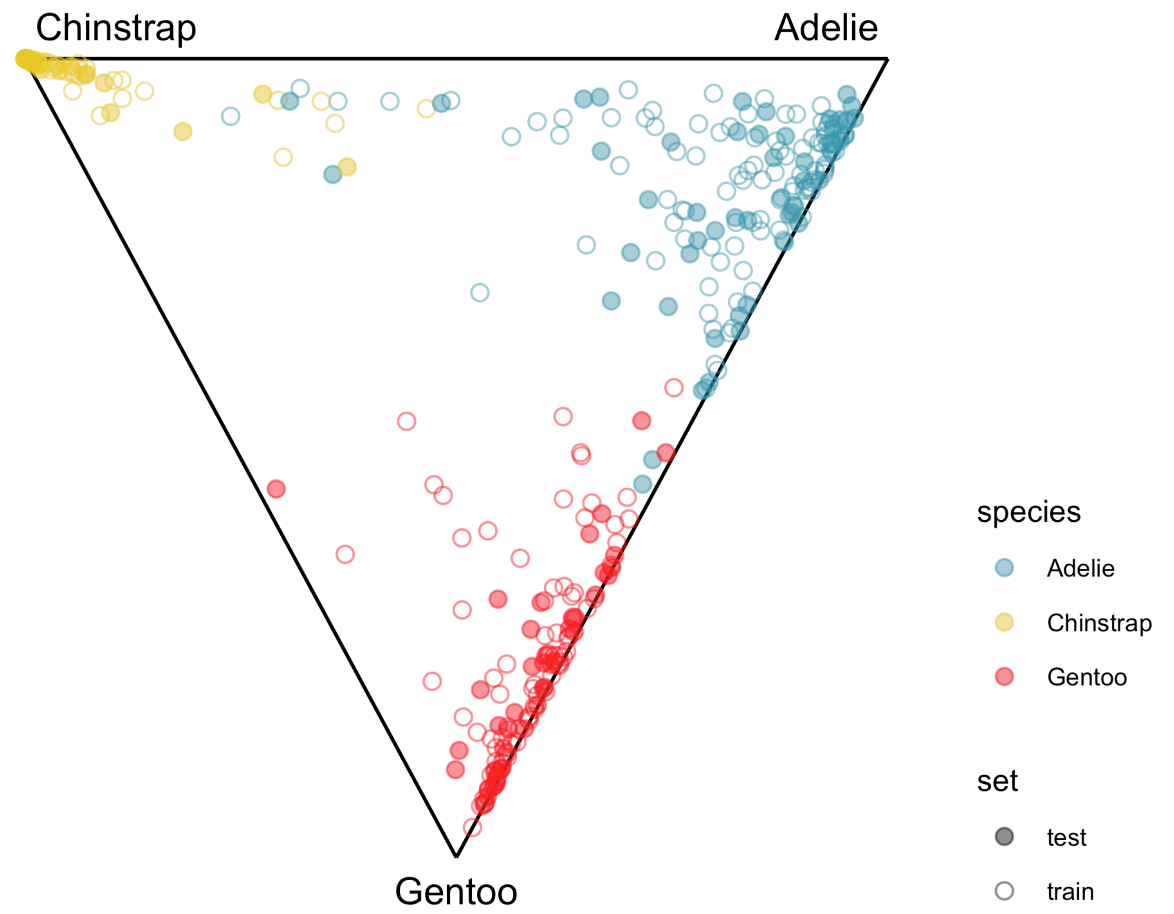
Check the fit at the hidden layer nodes



This is the dimension reduction induced by the model.

Realistically, with a complex neural network, it is too much work to check these nodes.

Examine the predictive probabilities



The problem with this model is that Gentoo are too confused with Adelie. This is a structural problem because from the visualisation of the 4D data we know that there is a big gap between the Gentoo and both other species.

Want to learn more?

- Work your way through the example of fitting the fashion MNIST data using tensorflow.
- Hands-on machine learning has a lovely step-by-step guide to constructing and fitting.
- This is a very nice slide set: A gentle introduction to deep learning in R using Keras
- And the tutorials at TensorFlow for R have lots of examples.

Next: Explainable artificial intelligence (XAI)